Data Science and Engineering (2019) 4:269–289
https://doi.org/10.1007/s41019-019-0097-5

Exploring the Semantic Content of Unsupervised Graph Embeddings: An Empirical Study

Stephen Bonner[1] · Ibad Kureshi[2] · John Brennan[1] · Georgios Theodoropoulos[3] · Andrew Stephen McGough[4] · Boguslaw Obara[1]

Received: 12 March 2018 / Revised: 23 May 2019 / Accepted: 22 June 2019 / Published online: 29 June 2019

Abstract
Graph embeddings have become a key and widely used technique within the field of graph mining, proving to be successful across a broad range of domains including social, citation, transportation and biological. Unsupervised graph embedding techniques aim to automatically create a low-dimensional representation of a given graph, which captures key structural elements in the resulting embedding space. However, to date, there has been little work exploring exactly which topological structures are being learned in the embeddings, which could be a possible way to bring interpretability to the process. In this paper, we investigate if graph embeddings are approximating something analogous to traditional vertex-level graph features. If such a relationship can be found, it could be used to provide a theoretical insight into how graph embedding approaches function. We perform this investigation by predicting known topological features, using supervised and unsupervised methods, directly from the embedding space. If a mapping between the embeddings and topological features can be found, then we argue that the structural information encapsulated by the features is represented in the embedding space. To explore this, we present extensive experimental evaluation with five state-of-the-art unsupervised graph embedding techniques, across a range of empirical graph datasets, measuring a selection of topological features. We demonstrate that several topological features are indeed being approximated in the embedding space, allowing key insight into how graph embeddings create good representations.

Keywords Graph embeddings · Neural networks · Representation learning

1 Introduction

Representing the complex and inherent links and relationships between and within datasets in the form of a graph is a widely performed practice across many scientific disciplines [1]. One reason for the popularity is that the structure or topology of the resulting graph can reveal important and unique insights into the data it represents. Recently, analysing and making predictions about graph using machine learning has shown significant advances in a range of commonly performed tasks over traditional approaches [2]. Such tasks include predicting the formation of new edges within the graph and the classification of vertices [3]. However, graphs are inherently complex structures and do not naturally lend themselves as input into existing machine learning methods, most of which operate on vectors of real numbers.

Graph embeddings[1] are a family of machine learning models which learn latent representations for the vertices within a graph. The goal of all graph embedding techniques is broadly the same: to transform a complex graph, with no inherent representation in vector space, into a low-dimensional vector (often in the range of 50–300) representation of the graph or its elements. More concretely, the objective of a graph embedding technique is to learn some function

✉ Stephen Bonner
s.a.r.bonner@durham.ac.uk

Boguslaw Obara
boguslaw.obara@durham.ac.uk

1 Department of Computer Science, Durham University, Durham, UK

2 InlecomSystems, Brussels, Belgium

3 School of Computer Science and Engineering, SUSTech, Shenzhen, China

4 School of Computing, Newcastle University, Newcastle, UK

[1] In this work, we focus on vertex representation learning approaches.

$f : V \rightarrow \mathbb{R}^d$ which is a mapping from the set of vertices V to a set of embeddings for the vertices, where d is the required dimensionality of the resulting embedding. This results in the mapping function f producing a matrix of dimensions $|V|$ by d, i.e. an embedding of size d for each vertex in the graph. It should be noted that this mapping is intended to capture the latent structure from a graph by mapping similar vertices together in the embedding space. Many of the recent approaches are able to produce low-dimensional graph representations without the need for labelled datasets. These representations can then be utilised as input to secondary supervised models for downstream prediction tasks, including classification [4] or link prediction [5]. Thus, unsupervised graph embeddings are becoming a key area of research as they act as a translation layer between the raw graph and some desired machine learning model.

However, to date, there has been little research performed into why graph embedding approaches have been so successful. They all aim to capture as much topological information as possible during the embedding process, but how this is achieved, or even exactly what structure is being captured, is currently not known. In this work, we focus solely upon unsupervised graph embedding techniques as we want to explore what features the techniques learn from the topology alone, without the requirement for labels. In previous work [6], we provided a framework which could be used to directly measure the ability of graph embeddings to capture a good representation of a graph's topology. In this paper, we expand upon this work by attempting to provide insight into the graph embedding process itself. We attempt to explore if the known and mathematically understood range of topological features [1] is being approximated in the embedding space. To achieve this, we investigate if a mapping from the embedding space to a range of topological features is possible. We hypothesise that if such a mapping can be found, then the topological structure represented by that feature is thus approximately captured in the embedding space. Such a discovery could start to provide a theoretical framework for the use of graph embeddings, by experimentally demonstrating which topological structures are approximated to create the representations.

Our methodology employs a combination of supervised and unsupervised models to predict topological features directly from the embeddings. The features we are investigating first go through a pre-processing step to transform them into discrete classes to enable classification. We make the following contributions whilst exploring this area:

- We investigate if unsupervised graph embeddings are learning something analogous with traditional vertex-level graph features. If this is the case, is there a particular type of feature which is being approximated most frequently?
- We empirically show, to the best of our knowledge for the first time, that several known topological features are present in graph embeddings. This can be used to help bring interpretability to the graph embedding process by detailing which graph features are key in creating high-quality representations.
- We provide detailed experimental evidence, with five state-of-the-art unsupervised graph embedding approaches, across seven topological features and six empirical graph datasets to support our claims.

Reproducibility—we make all experiments performed in this paper reproducible by open-sourcing our code,[2] reporting key model hyper-parameters and presenting results on public benchmark graph datasets.

In Sect. 2 we explore prior work. In Sect. 3 we detail our approach for providing an experimental methodology for assessing known topological features approximated by graph embeddings; Sect. 4 details the experiment set-up. In Sect. 5 we present our results, and in Sect. 6 we conclude this paper along with suggestions for further expansions of this work.

1.1 Notation

We adopt here the commonly used notation for representing a graph or network[3] $G = (V, E)$ as an undirected graph which comprises a finite set of vertices (sometimes referred to as nodes) V and a finite set of edges E. The elements of E are an unordered tuple $\{u, v\}$ of vertices $u, v \in V$. Here G could be a graph-based representation of a social, citation or biological network [7]. The adjacency matrix $\mathbf{A} = (a_{i,j})$ for a graph is symmetric matrix of size $|V|$ by $|V|$, where $(a_{i,j})$ is 1 if an edge is present between vertices i and j or 0 otherwise.

2 Previous Work

This section explores the prior research regarding graph embedding techniques and previous approaches measuring known features in embeddings. We first introduce the notation of graph embeddings, detail supervised and factorisation-based approaches, explore in detail state-of-the-art unsupervised approaches which will be used throughout the rest of the paper and finally review past attempts to provide interpretability to embedding approaches.

[2] https://github.com/sbonner0/unsupervised-graph-embedding/.

[3] To avoid confusion with neural networks we will use the term graph throughout the remainder of the paper without loss of generality.

2.1 Introduction to Graph Embeddings

The ability to automatically learn some descriptive numerical-based representation for a given graph is an attractive goal and could provide a timely solution to some common problem within the field of graph mining. Traditional approaches have relied upon extracting features—such as various measures of a vertex's centrality [8]—capturing the required information about a graph's topology, which could then be used in some downstream prediction task [9–12]. However, such a feature extraction-based approach relies solely upon the handcrafted features being a good representation of the target graph. Often a user must use extensive domain knowledge to select the correct features for a given task, with a change in task often requiring the selection of new features [9].

Graph embedding models are a collection of machine learning techniques which attempt to learn key features from a graph's topology automatically, in either a supervised or unsupervised manner, removing the often cumbersome task of end-users manually selecting representative graph features [4]. This process, known as feature selection [13] in the machine learning literature, has clear disadvantages as certain features may only be useful for a certain task. It could even negatively affect model performance if utilised in a task for which they are not well suited. Arguably, many of the recent exciting advances seen in the field of deep learning have been driven by the removal of this feature selection process [5], instead of allowing models to learn the best data representations themselves [14]. For a selection of recent review papers covering the complete family of graph embedding techniques, readers are referred to [15–18]. The work presented in this paper focuses on neural network-based approaches for graph embedding (as these have demonstrated superior performance compared with traditional approaches [2]).

The study of neural networks (NNs) is a field within machine learning inspired by the brain [14]. NNs model problems via the use of connected layers of artificial neurons, where each network has an input layer, at least one hidden layer and an output layer. The activation of each neuron in a layer is given by a pre-specified function, with each neuron taking a weighted sum of all the outputs of those neurons which feed into it. These weights are learned through training examples which are fed through the network, with modifications made to the weights via back-propagation to increase the probability of the NN producing the desired result [14].

2.1.1 Supervised Approaches

Within the field of machine learning, approaches which are supervised are perhaps the most studied and understood [14]. In supervised learning, the datasets contain labels which help guide the model in the learning process. In the field of graph analysis, these labels are often present at the vertex level and contain, for example, the metadata of a user in a social network.

Perhaps the largest area of supervised graph embeddings is that of graph convolutional neural networks (GCNs) [19], both spectral [20, 21] and spatial [22] approaches. Such approaches pass a sliding window filter over a graph, in a manner analogous with convolutional neural networks from the computer vision field [14], but with the neighbourhood of a vertex replacing the sliding window. Current GCN approaches are supervised and thus require labels upon the vertices. This requirement has two significant disadvantages: Firstly, it limits the available graph data which can be used due to the requirement for labelled vertices. Secondly, it means that the resulting embeddings are specialised for one specific task and cannot be generalised for a different problem without costly retraining of the model for the new task.

2.1.2 Factorisation Approaches

Before the recent interest in learning graph embeddings via the use of neural networks, a variety of other approaches were explored. Often these approaches took the form of matrix factorisation, in a similar vein to classical dimensionality reduction techniques such as principal competent analysis (PCA) [15, 23]. Such approaches first calculate the pairwise similarity between the vertices of a graph and then find a mapping to a lower-dimensional space such that the relationships observed in the higher dimensions are preserved. An early example of such an approach is that of the Laplacian eigenmaps, which attempts to directly factorise the Laplacian matrix of a given graph [24]. Other approaches, often using the adjacency matrix, define the relationship in low-dimensional space between two vertices in the graph as being determined by the dot-product of their corresponding embeddings. Such approaches include graph factorisation [25], GraGrep [26] and HOPE [27]. Such dimensionality reduction-based approaches are often quadratic in complexity [17], and the predictive performance of the embeddings has largely been superseded by the recent neural network-based methods [2].

2.2 Unsupervised Stochastic Embeddings

DeepWalk [4] and Node2Vec [5] are the two main approaches for random walk-based embedding. Both of these approaches borrow key ideas from a technique entitled Word2Vec [28] designed to embed words, taken from a sentence, into vector space. The Word2Vec model is able to learn an embedding for a word by using surrounding words within a sentence as targets for a single hidden layer

neural network model to predict. Due to the nature of this technique, words which co-occur together frequently in sentences will have positions which are close within the embedding space. The approach of using a target word to predict neighbouring words is entitled Skip-Gram and has been shown to be very effective for language modelling tasks [29].

2.2.1 DeepWalk

The key insight of DeepWalk is to use random walks upon the graph, starting from each vertex, as the direct replacement for the sentences required by Word2Vec. A random walk can be defined as a traversal of the graph rooted at a vertex $v_t \in V$, where the next step in the walk is chosen uniformly at random from the vertices incident upon v_t [30]; these walks are recorded as $w_0^t, \ldots, w_n^t$ (where t is the walk starting from v_t of length n, and $w_i^t \in V$), i.e. a sequence of the vertices visited along the random walk starting from $v_t = w_0^t$. DeepWalk is able to learn unsupervised representations of vertices by maximising the average log probability $\mathbf{P}$ over the set of vertices V:

$$\frac{1}{|V|} \sum_{t=1}^{|V|} \sum_{i=0}^{n} \sum_{-c \le j \le c, j \neq 0} \log \mathbf{P}(w_{i+j}^t | w_i^t), \tag{1}$$

where c is the size of the training context of vertex w_n^t.[4]

The basic form of Skip-Gram used by DeepWalk defines the conditional probability $\mathbf{P}(w_{i+j}^t | w_i^t)$ of observing a nearby vertex w_{i+j}^t, given the vertex w_i^t from the random walk t, can be defined via the softmax function over the dot-product between their features [4]:

$$\mathbf{P}(w_{i+j}^t | w_i^t) = \frac{\exp\left(\mathbf{W}_{w_i^t}^{\intercal} \mathbf{W}'_{w_{i+j}^t}\right)}{\sum_{t=1}^{|V|} \exp\left(\mathbf{W}_{w_i^t}^{\intercal} \mathbf{W}'_{v_t}\right)}, \tag{2}$$

where $\mathbf{W}_{w_i^t}$ and $\mathbf{W}'_{w_{i+j}^t}$ are the hidden layer and output layer weights of the Skip-Gram neural network, respectively.

2.2.2 Node2Vec

Whilst DeepWalk uses a uniform random transition probability to move from a vertex to one of its neighbours, Node2Vec biases the random walks. This biasing introduces two user-controllable parameters which dictate how far from, or close to, the source vertex the walk progresses. This is done to capture either the vertex's role in its local neighbourhood (homophily), or alternatively, its role in the global graph structure (structural equivalence) [5]. Changing the random walk means that Node2Vec has a higher accuracy over DeepWalk for a selection of vertex classification problems [5].

2.3 Unsupervised Hyperbolic Embeddings

Recently, a new family of graph embedding approaches has been introduced, which embed vertices into hyperbolic, rather than Euclidean space [31, 32]. Hyperbolic space has long been used to analyse graphs which exhibit high levels of hierarchical or community structure [33], but it also has properties which could make it an interesting space for embeddings [32]. Hyperbolic space can be considered 'larger' than Euclidean with the same number of dimensions; as the space is curved, its total area grows exponentially with the radius [32]. For graph embeddings, this key property means that one effectively has a much larger range of possible points into which the vertices can be embedded. This property allows for closely correlated vertices to be embedded close together, whilst also maintaining more distance between disparate vertices, resulting in an embedding which has the potential to capture more of the latent community structure of a graph.

The hyperbolic approach we focus on was introduced by Chamberlain [32] and uses the Poincaré disc model of 2D hyperbolic space [34]. In their model, the authors use polar coordinates $x = (r, \theta)$, where $r \in [0, 1]$ and $\theta \in [0, 2\pi]$ to describe a point in space for each vertex v in the Poincaré disc, which allows for the technique to be significantly simplified [32]. Similar to DeepWalk, an inner-product is used to define the similarity between two points within the space. The inner-product of two vectors in a Poincaré disc can be defined as follows [32]:

$$< x, y >= ||x|| ||y|| \cos(\theta_x - \theta_y), \tag{3}$$

$$= 4 \operatorname{arctanh} r_x \operatorname{arctanh} r_y \cos(\theta_x - \theta_y), \tag{4}$$

where $x = (r_x, \theta_x)$ and $y = (r_y, \theta_y)$ are the two input vectors representing two vertices and *arctanh* is the inverse hyperbolic tangent function [32].

To create their hyperbolic graph embedding, the authors use the softmax function of Eq. 2, common with DeepWalk and others, but importantly replacing the Euclidean inner-products with the hyperbolic inner-products of Eq. 3. Aside from this, hyperbolic approaches share many similarities with the stochastic approaches with regard to their input data and training procedure. For example, the hyperbolic approaches are still trained upon pairs of vertex IDs, taken from sequences of vertices generated via random walks on graphs.

2.4 Unsupervised Auto-encoder-Based Approaches

There is an alternative set of approaches for graph embeddings which do not rely upon random walks. Instead of

[4] Note if $i + j < 0$ then we skip these from the sum as we are past the start of the current walk.

adapting a technique based upon capturing the meaning of language, such models are designed specifically for creating graph embeddings using deep learning [14]—deep auto-encoders [35]. Auto-encoders are an unsupervised neural network, where the goal is to accurately reconstruct the input data through explicit encoder and decoder stages [36]. Two such approaches are structural deep network embedding (SDNE) [37] and deep neural networks for learning graph representations (DNGR) [38].

The authors of these approaches argue that a deep neural network, versus the shallow Skip-Gram model used by both DeepWalk and Node2Vec, is much more capable of capturing the complex structure of graphs. In addition, the authors argue that for a successful embedding, it must capture both the first- and second-order proximity of vertices. Here the first-order proximity measures the similarity of the vertices which are directly incident upon one another, whereas the second-order proximity measures the similarity of vertices neighbourhoods. To capture both of these elements SDNE has a dual-objective loss function for the model to optimise. The input data to SDNE are the adjacency matrix $\mathbf{A}$, where each row a represents the neighbourhood of a vertex.

The objective function for SDNE comprises two distinct terms: The first term captures the second-order proximity of the vertices neighbourhood, whilst the second captures the first-order proximity of the vertices by iterating over the set of edges E:

$$L_{\text{SDNE}} = \sum_{i=1}^{|V|} ||(q_i' - q_i) \odot b_i||_2^2 + \alpha \sum_{u,v=1}^{|E|} a_{u,v} ||(w_u^{(k)} - w_v^{(k)})||_2^2, \tag{5}$$

where q_i and q_i' are the input and reconstructed representation of the input, $\odot$ is the element-wise Hadamard product, and b_i is a scaling factor to penalise the technique if it predicts zero too frequently, $w^{(k)}$ is the weight of the kth layer in the auto-encoder technique, and α is a user-controllable parameter defining the importance of the second term in the final loss score [37].

To initialise the weights of the deep auto-encoder used for this approach, an additional neural network must be trained to find a good starting region for the parameters. This pre-training neural network is called a deep belief network and is widely used within the literature to form the initialisation step of deeper models [39]. However, this pre-training step is not required by either the stochastic or hyperbolic approaches as random initialisation is used for the weights and adds significant complexity.

In comparison with SDNE, instead of relying solely upon the raw adjacency matrix as input, DNGR creates a new denser representation to be passed to an auto-encoder [38]. The authors have the model reconstruct the pointwise mutual information matrix (PPMI) of the input graph, which captures vertex co-occurrence information in a sequence created via a random surfer model. Additionally, instead of passing this to a traditional auto-encoder, a stacked de-noising auto-encoder is used with the goal of creating a more robust vertex representation. This stacked de-noising auto-encoder adds a small quantity of noise to the input data, which the model must learn to disregard during the training process.

2.5 Observing Features Preserved in Embeddings

2.5.1 Graph Embeddings Features

To date, there has been little research performed exploring a theoretical basis as to why graph embeddings are able to demonstrate such good performance in graph analytic tasks, or if something approximating traditional graph features are being captured during the embeddings process. Goyal and Ferrar [2] presented an experimental review paper on a selection of graph embedding techniques. The authors use a range of tasks including vertex classification, link prediction and visualisation to measure the quality of the embeddings. However, the authors do not provide any theoretical basis as to why the embedding approaches they test are successful, or if known features are present in the embeddings. In addition, the authors do not consider embeddings taken from promising unsupervised techniques, such as the family of hyperbolic approaches, nor do they explore performance across imbalanced classes during the classification.

Recent work has speculated on the use of a graph's topological features as a way to improve the quality of vertex embeddings by incorporating them into a supervised GCN-based model [40]. They show how aggregating a vertex feature—even one as simple as its degree—can improve the performance of their model. Further, they present theoretical analysis to validate that their approach is able to learn the number of triangles a vertex is part of, arguing that this demonstrates the model is able to learn topological structure. We take inspiration from this work, but consider unsupervised approaches as well as exploring if richer and more complicated topological features are being captured in the embedding process. In a similar vein, an approach for generating supervised graph embeddings using heat-kernel-based methods is validated by visualizing if a selection of topological features is present in a two-dimensional projection of the embedding space [41].

Research has investigated the use of a graph's topological features as a way of validating the accuracy of a neural network-based graph generative model [42]. With the presented model, the authors aim to generate entirely new graph datasets which mimic the topological structure of a set of target graphs—a common task within the graph mining community [43]. To validate the quality of their model,

they investigate if a new graph created from their generative procedure has a similar set of topological features to the original graph.

Perhaps most closely related to our present research is work exploring the use of random walk-based graph embeddings as an approximation for more complex vertex-level centrality measures on social network graphs [44]. The authors argue that graph embeddings could be used as a replacement for centrality measures as they potentially have a lower computational complexity. The work explores the use of linear regression to try to directly predict four centrality measures from the vertices of three graph datasets, with limited success [44]. Our own work differs significantly as we attempt to provide insight into what exactly graph embeddings are learning with a view to allow for greater interpretability, explore a wider range of embeddings approaches, use datasets from a wider range of domains, explore more topological features, use classification rather than regression as the basis for the analysis and address the inherent unbalanced nature of most graph datasets.

2.5.2 Feature Learning in Other Domains

A large quantity of the successful unsupervised graph embedding approaches have adapted models originally designed for language modelling [4, 5]. Some recent research investigated how best to evaluate a variety of unsupervised approaches for embedding words into vectors [45]. They choose a variety of natural language processing (NLP) tasks, which capture some known and understood aspects of the structure of language, and investigate how well the chosen embedding models perform for these tasks. They conclude that no single word embedding model performs the best across all the tasks they investigated, suggesting there is not a single optimal vector representation for a word. What features are used to help word embeddings achieve compositionality, constructing the meaning of an entire sentence from the component words, has also been explored [46]. Further research has investigated the use of word embeddings to create representations for the entire sentence using word features [47]. The work suggests that word features learned by the embeddings for natural language inference can be transferred to other tasks in NLP.

Outside of NLP, there has been work in the field of computer vision (CV) investigating what known features, already commonly used for image representation, are captured by deep convolutional neural network—potentially being used to explain how they work. For example, it has been shown that convolutional networks, when trained for image classification, often detect the presence of edges in the images [48]. The same work also shows how the complexity of the detected edges increases as the depth of the network increases.

In this present work, we take inspiration from these approaches and attempt to provide insight and a potential theoretical basis for the use of graph embeddings by exploring which known graph features can be reconstructed from the embedding space.

3 Semantic Content of Graph Embeddings

Despite extensive prior work in unsupervised graph embedding highlighting how they perform well for the tasks for which they were proposed (such as vertex classification and link prediction [2]), there has been little work in exploring why these approaches are successful. This could allow for an increased level of interpretability to graph embeddings. Inspired by recent work in computer vision and natural language processing which examine if traditional features (the edges detected in images for example) are captured by deep models, we explore the following research question:

Problem Statement Do unsupervised graph embedding techniques capture something similar to traditional topological features as part of the embedding process?

Topological features are one known and mathematically understood way to accurately identify graphs and vertices [9, 10]. We hypothesise that if graph embeddings are shown to be learning approximations of existing features, this could begin to provide a theoretical basis for the interpretability of graph embeddings. This would suggest that graph embeddings are automatically learning detailed and known graph structures in order to create the representations. This could explain how they have been so successful in a variety of graph mining tasks. Effectively, the graph embedding techniques would be acting as an automated way of learning the most representative topological feature(s) for a given objective function.

If graph embeddings are shown to be learning topological features, then other interesting research questions arise. For example, do competing embedding approaches learn different topological structures, do different graph datasets each require different features to be approximated in order to create a good representation, what is the structural complexity of the features approximated by the embeddings, or even are the embeddings capable of approximating multiple features simultaneously?

In order to explore these questions, we attempt to predict a selection of topological features directly from graph embeddings computed from a range of state-of-the-art approaches across a series of empirical datasets. We suggest that if a second mapping function $f : \mathbb{R}^d \rightarrow \Lambda$ can be found which accurately maps the embedding space to a given topological feature Λ, then this is strong evidence that something approximating the structural information represented by Λ is indeed present in the embedding space.

Here the mapping function could take the form of a linear regression, but for this work, we investigate a range of classification algorithms—this is explored more in Sect. 3.3. We assess a range of known topological features, from simple to complex, to gain a better understanding of the expressive capabilities of the embedding techniques.

3.1 Predicting Topological Features

Numerous topological features have been identified in the literature, measuring various aspects of a graph's topology, at the vertex, edge and graph levels [9]. As we are focusing our work here upon methods for creating vertex embedding, we will focus on features which are measured at the *vertex level* of a given graph. We have selected a range of vertex-level features from the graph mining literature, which capture information about a vertex's local and global role within a graph [5]. This selection of features ranges from ones which are simple to compute from vertices directly adjacent to each other, to more complex features which can require information from many hops[5] further along with the graph. This will allow us to explore if embedding models learn complex topological features, or are they able to learn good representations of only simple features. The topological vertex-level features we are predicting are detailed below, listed approximately by their complexity:

These features are defined in terms of a graph $G = (V, E)$ with its corresponding adjacency matrix $\mathbf{A}$, where $|V|$ is the total number of vertices in the graph and $|E|$ the total number of edges. For each vertex $v \in V$, we also define $d(v)$ to be the total number of neighbours for v, $d^+(v)$ to be the number of connections v has to other vertices, $\Gamma^-(v)$ to be the subset of vertices in V with edges to v, and $\sigma_{st}(v)$ is the total number of shortest paths from vertices s and t which also pass through v.

Total degree $\mathrm{DG}(v) = d(v)$: the total number of edges from v to other vertices.

Degree centrality $\mathrm{DC}(v) = \frac{1}{|V|} d(v)$: the degree for the vertex v over the total number of vertices in the graph, providing a normalised centrality score [10].

Number of triangles $\mathrm{TC}(v) = \Phi$: the number of triangles containing the vertex v, where Φ is the number of vertices in $\Gamma^-(v)$ which are also connected via an edge [10].

Local clustering score $\mathrm{CLU}(v) = \frac{2\Phi}{d(v)(d(v)-1)}$: represents the probability of two neighbours of v also being neighbours of each other [49].

Eigenvector centrality $\mathrm{EC}(v) = \mathbf{Ax} = \lambda\mathbf{x}$: used to calculate the importance of each vertex within a graph, where λ is the largest eigenvalue and $\mathbf{x}$ is the eigenvector centrality [50].

PageRank centrality $\mathrm{PR}(v) = \frac{1-\Omega}{|V|} + \Omega \sum_{u \in \Gamma^-(v)} \frac{PR(u)}{d^+(u)}$: PageRank centrality is commonly used to measure the local influence of a vertex within a graph [8, 51], where Ω is a constant damping factor (0.85 for this work).

Betweenness centrality $\mathrm{BC}(v) = \sum_{\substack{s \neq v \neq t \in V \\ s \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}}$: The betweenness centrality of a vertex depends upon the frequency which acts as a bridge between two additional vertices [51], where σ_{st} is the total number of shortest paths from s to t.

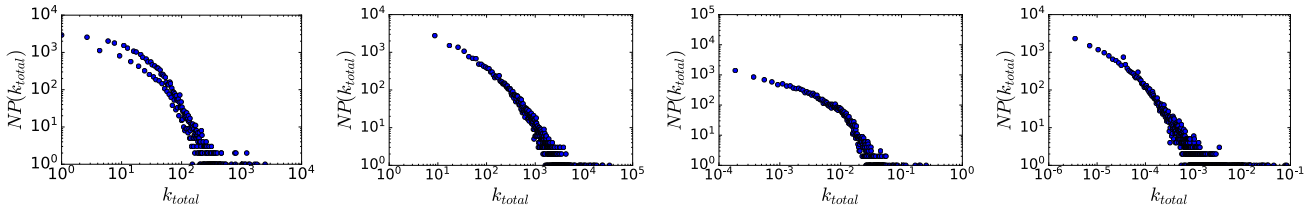
3.2 Power-Law Feature Distribution

Many empirical graphs, especially those representing social, hyperlink and citation networks, have been shown to have approximately a power-law distribution of degree values [52]. This power-law distribution poses a challenge for machine learning models, as it means the features we are trying to predict are extremely unbalanced, with a heavy skew towards the lower range of features. Imbalanced class distribution creates difficulties for machine learning models, as there are fewer examples of the minority classes for the model to learn, which can often lead to poor predictive performance on these classes [14]. It has been shown that the distribution of other topological features can also follow a power-law distribution in many graphs [43]. To demonstrate this phenomenon, Fig. 1 shows the distribution of a range of topological feature values for the cit-HepTh dataset. The figure shows that indeed, all the topological feature values tested largely follow an approximately power-law distribution. This fact has the potential to make predicting the value of a certain topological feature challenging, as the datasets will not be balanced and any model attempted to find the mapping $f : \mathbb{R}^d \rightarrow \Lambda$ will be prone to over-fitting to the majority classes. Our approach for tackling this issue is outlined in the following section.

3.3 Methodology

Unlike previous studies [44] we employ classification and visualisation, instead of regression, as ways to explore the embedding space. We chose these approaches as predicting topological features directly via the use of regression has proven challenging in prior work [44], owing largely to the imbalance problem explored in the previous section. With such an imbalanced dataset, using a classification-based approach is often advantageous [53] as techniques exist to over-sample minority examples. However, the features we are attempting to predict are continuous, so must go through some transformation stage before classification can be performed. For our transformation stage, we follow a procedure

[5] Hops represent the length of the sequences of vertices that must traversed to get from vertices i to j.



(a) Degree **(b)** Triangle Count **(c)** Eigenvector Cent **(d)** Betweenness Cent

Fig. 1 Distribution of topological feature values from the cit-HepTh dataset in log scale: **a** total vertex degree distribution, **b** distribution of complete triangles for each vertex, **c** eigenvector centrality distribution and **d** betweenness centrality score distribution

similar to that introduced by Oord et al. [53]. We bin the real-valued features into a series of classes via the use of a histogram, where the bin in which a particular feature is placed becomes its class label. One can consider each of these newly created classes as representing a range of possible values for a given feature. As an example, we could transform a vertex's continuous PageRank score [8] into a series of discrete classes via the use of a histogram with a bin size of three, where each of the newly created classes represented a low, medium or high PageRank score.

Although this binning process helps with the feature imbalance, it still produces a skew in number of features assigned to each class. To further address this issue, we take the logarithm of each feature value before it is passed to the binning function. Essentially, this will mean that features within the same order of magnitude will be assigned the same class; for example, vertices with degrees in the range of 0–10^1 would be assigned into one class, whilst degree values between 10^2 and 10^3 would be assigned to another class. This was performed as it dramatically improved the balance of the datasets, and as we are only attempting to discover if something approximating the topological features is present in the embedding space, we found that predicting the order of magnitude to be sufficient.

In order to allow for a good distribution of feature values in the datasets we are using, in our experiments we utilise a bin size of six for the histogram function, meaning that six discrete classes were created for each of the features. This value was chosen empirically from our datasets as it fully covered the numerical range of the topological features we measured. For example, we found that the centrality values in our datasets fall within a range of six orders of magnitude, which is what we used to set the number of bins. It should be noted that this value would need to be tuned depending upon the datasets and features being used.

In addition to the use of classification, we explore an additional method to bring interpretability to graph embeddings, that being a visualisation technique entitled t-SNE [54]. This technique allows relatively high-dimensional data, such as graph embeddings, to be projected into a low-dimensional space in such a way as to preserve the inter-spatial relationship between points that were present in the original space. Thus, we utilise t-SNE to project the embeddings down to two dimensions, so they can be easily visualised. This process is performed without the need for any classification to be trained upon the embeddings, removing the issues associated with classifying unbalanced datasets. Once the projection has been performed, we can colour each point in accordance with its feature value, be that one that has been transformed via the binning process, or even the raw value itself.

Table 1 Graph embedding approaches being compared

Approach	Year	Type	Published	Complexity
DeepWalk	2014	Stochastic	KDD [4]	$O(\|V\|)$
Node2Vec	2016	Stochastic	KDD [5]	$O(\|V\|)$
SDNE	2016	Auto-encoder	KDD [37]	$O(\|V\|\|E\|)$
Poincaré disc	2017	Hyperbolic	MLG [32]	$O(\|V\|)$

3.4 Embedding Approaches Compared

In this paper, we evaluate five state-of-the-art unsupervised graph embedding approaches as a way of exploring what semantic content is extracted from a graph to create the embeddings. The approaches are as follows: DeepWalk, Poincaré disc, structural deep network embedding and Node2Vec,[6] which are detailed in Table 1. These approaches were chosen as they represent a good cross section of the current competing methodologies and all either exploit a different method of sampling the graph, use different geometries for the embedding space or use competing methods of comparing vertices. This selection of approaches will allow exploration of interesting research questions. Such questions include if any differences between the approaches can be explained by what graph structures they learn and do methods which promote local

[6] Please note that we explore two variations of Node2Vec, bringing the total number of approaches to five.

exploration around the target vertex only learn local structural information, degree for example. To explore this second question in more detail, we created two versions of Node2Vec: Node2Vec-Structural, which biases the random walks used to create training pairs for the model to explore vertices further away from the target vertex, and Node2Vec-Homophily, which biases the random walks to stay closer to the target vertex.

4 Experimental Set-up and Classification Algorithm Selection

In the following section we detail the set-up of the experiments and evaluate potential classification algorithms.

4.1 Metrics

4.1.1 Presented Results

All the reported results are the mean of five replicated experiment runs along with confidence intervals. For the run-time analysis, the presented results are the mean run-time for job completion, presented in minutes. For the classification results, all the accuracy scores presented are the mean accuracy after k-fold cross-validation—considered the gold standard for model testing [55]. For k-fold cross-validation, the original dataset is partitioned into k equally sized partitions. $k-1$ partitions are used to train the model, with the remaining partition being used for testing. The process is repeated k times using a unique partition for each repetition and a mean taken to produce the final result.

4.1.2 Precision Metrics

For reporting the results of the vertex feature classification tasks, we report the macro-f1 and micro-f1 scores with varying percentages of labelled data available at training time. This is a similar set-up to previous works [2, 5].

The micro-f1 score calculates the f1-score for the dataset globally by counting the total number of true positives (TP), false positives (FP) and false negatives (FN) across a labelled dataset $|L|$. Using the notation from [2], micro-f1 is defined as:

$$\text{microf1} = \frac{2 \cdot P \cdot R}{P + R}, \tag{6}$$

where

$$\text{Precision}(P) = \frac{\sum_{l=1}^{|L|} \text{TP}(l)}{\sum_{l=1}^{|L|} \text{TP}(l) + \text{FP}(l)},$$

$$\text{Recall}(R) = \frac{\sum_{l=1}^{|L|} \text{TP}(l)}{\sum_{l=1}^{|L|} \text{TP}(l) + \text{FN}(l)},$$

and $\text{TP}(l)$ denotes the number of true positives the model predicts for a given label l, $\text{FP}(l)$ denotes the number of false positives, and $\text{FN}(l)$ denotes the number of false negatives.

The macro-f1 score, when performing multi-label classification, is defined as the average micro-f1 score over the whole set of labels L:

$$\text{macrof1} = \frac{1}{|L|} \sum_{l \in L} \text{micro-f1}(l), \tag{7}$$

where $\text{microf1}(l)$ is the micro-f1 score for the given label l.

4.2 Experimental Set-up

4.2.1 Implementation Details

The approaches used for experimentation were re-implemented in TensorFlow [56], as the author-provided versions were not all available using the same framework. We also ensure the same TensorFlow-based optimisations were used across all the approaches wherever possible [57]. Neural Networks contain many hyper-parameters a user can control to improve the performance, both of the predictive accuracy and the run-time, of a given dataset. This process can be extremely time consuming and often requires users to perform a grid search over a range of possible hyper-parameter values to find a combination which performs best [14]. For setting the required hyper-parameters for the approaches, we used the default hyper-parameters as proposed by the authors in their original papers, keeping them constant across all datasets. The key hyper-parameters used for each approach are detailed in Table 2. We have open-sourced our implementations of these approaches and made them available online.[7]

4.2.2 Experimental Environment

Experimentation was performed on a compute system with 2 NVIDIA Tesla K40c's, 2.3 GHz Intel Xeon E5-2650 v3, 64 GB RAM and the following software stack: Ubuntu Server 16.04 LTS, CUDA 9.0, CuDNN v7, TensorFlow 1.5, scikit-learn 0.19.0, Python 3.5 and NetworkX 2.0.

4.2.3 Experimental Datasets

The empirical datasets used for evaluation were taken from the Stanford Network Analysis Project (SNAP) data repository [58] and the Network Repository [59] and are detailed in Table 3. The domain label provided is taken from the

[7] https://github.com/sbonner0/unsupervised-graph-embedding/.

Table 2 Key hyper-parameter settings

Approach	Optimiser	Learning rate	Specific parameters
SDNE	RMSProp	0.01	$\alpha = 500$, $b = 10$, epochs = 500
Node2Vec-S	SGD	0.1	$p = 0.5$, $q = 2$, epochs = 15
Node2Vec-H	SGD	0.1	$p = 1.0$, $q = 0.5$, epochs = 15
DeepWalk	SGD	0.1	epochs = 15
Poincaré disc (PD)	SGD	0.1	$p = 0.5$, $q = 2$, epochs = 15

Table 3 Empirical graph datasets

Dataset	$\|V\|$	$\|E\|$	Domain	Source
Fly-drosophila-medulla	1800	33,500	Biological	[59]
Cit-HepTh	27,770	352,807	Citation	[58]
Email-Eu-core	1005	25,571	Communication	[58]
Inf-openflights	2900	30,500	Infrastructure	[59]
Soc-sign-bitcoinotc	5881	35,592	Blockchain	[58]
Ego-Facebook	4039	88,234	Social	[58]

listings of the graphs domain provided by SNAP [58] and Network Repository [59].

4.3 Classification Algorithm Selection

As highlighted throughout the paper, we are focusing our research on unsupervised graph embedding approaches. In order to be able to use the embeddings for further analysis, they must be classified using a supervised classification model. Traditionally in the embedding literature, a simple logistic regression is used in any classification task [4, 28], with seemingly little work exploring the use of more sophisticated models to perform the classification.

In this section we explore the effectiveness of five different models at performing the classification of the different embedding approaches—logistic regression (LR), support vector machine (SVM) (linear kernel), SVM (RBF kernel), a single hidden layer neural network and finally a second more complex neural network with two hidden layers and a larger number of hidden units. All the classifiers utilised in this section were taken from the Scikit-Learn Python package [60]. Additionally, given that our datasets do not have a equal distribution among the classes, we also explore the effectiveness of weighting the loss function used by the model inversely proportional to the frequency of the class [61]. This use of a weighted loss function, although common in other areas of machine learning, has not been explored in regard to graph embeddings.

For the results in this section, we present the mean macro- and micro-f1 scores, introduced in Sect. 4.1.2, after fivefold cross-validation. To assess the performance of the classifiers against the imbalance present in the datasets, we also display the percentage lift in mean test set accuracy over three rule-based prediction methods to act as baselines. These methods are uniform prediction (where the classification of each item in the test is chosen uniformly at random from the possible classes), stratified prediction (where the classification follows the distribution of classes in the training set) and frequent class prediction (where the classification is determined by the most frequency class in the training set). A positive lift across all metrics strongly suggests that a mapping from the embedding space to the topological features is being learned, as the classification algorithm is overcoming the biased distributions of classes in the dataset.

We performed this experiment for all combinations of datasets, embedding approaches and features, but due to the large quantity of results, we present only a subset here. Specifically, we present the results for ego-Facebook dataset, using embeddings generated by DeepWalk and SDNE and classifying degree, triangle count and eigenvector centrality. It should be noted that the patterns displayed here are representative of ones seen across all datasets.

Table 4 highlights the performance of the potential classifiers, when using the DeepWalk embeddings taken from the ego-Facebook dataset. Results show that the choice of supervised classifier can have a large impact on the overall classification score. It can also be seen that the traditional choice of logistic regression does not produce the best results. Indeed, the neural network and SVM classifier often gave the best scores, but no single classifier is best overall, suggesting that one needs to be chosen carefully for a given task.

Table 5 highlights the results for the potential classifiers, when using the SDNE embeddings taken from the ego-Facebook dataset. Again, the variation in classification score across the set of tested classification metrics is quite substantial, with the linear SVM and neural network approaches having perhaps a small margin of improvement over the others. It is interesting to note that the logistic regression frequently used in the literature never has the highest score in any metric. It can also be seen that, when compared to the DeepWalk results in Table 4, SDNE does less well at predicting all topological features which, although not the explicit purpose of this section, is interesting to note.

Using the results from this section, particularly the generally higher macro-f1 scores which indicate a better

Table 4 Degree (DG), triangle count (TC) and eigenvector centrality (EC) classification results for DeepWalk embeddings on the ego-Facebook dataset

Feature	Classifier	Micro-f1	Macro-f1	Uniform	Strat	Freq
DG	LR	0.336 (±0.015)	0.190 (±0.012)	+65.09%	+33.85%	+12.07%
	SVM (Lin)	**0.339 (±0.017)**	0.164 (±0.013)	**+66.57%**	**+35.03%**	**+13.07%**
	SVM (RBF)	0.336 (±0.021)	0.158 (±0.013)	+65.09%	+33.84%	+12.07%
	NN	0.329 (±0.013)	**0.200 (±0.018)**	+61.65%	+31.05%	+9.73%
	NN-2	0.326 (±0.016)	0.192 (±0.019)	+60.18%	+29.85%	+8.73%
TC	LR	0.340 (±0.011)	0.154 (±0.014)	+109.34%	+37.19%	+12.38%
	SVM (Lin)	**0.344 (±0.015)**	0.139 (±0.006)	**+111.8%**	**+38.8%**	**+13.7%**
	SVM (RBF)	0.335 (±0.018)	0.130 (±0.010)	+106.26%	+35.17%	+10.73%
	NN	0.331 (±0.019)	0.157 (±0.013)	+103.8%	+33.56%	+9.4%
	NN-2	0.326 (±0.017)	**0.163 (±0.015)**	+100.72%	+31.54%	+7.75%
EC	LR	0.590 (±0.013)	0.474 (±0.010)	+195.66%	+144.16%	+92.18%
	SVM (Lin)	0.591 (±0.012)	0.480 (±0.011)	+196.16%	+144.58%	+92.51%
	SVM (RBF)	0.552 (±0.012)	0.446 (±0.011)	+176.62%	+128.44%	+79.8%
	NN	0.629 (±0.012)	0.512 (±0.017)	+215.2%	+160.3%	+104.89%
	NN-2	**0.630 (±0.019)**	**0.513 (±0.021)**	**+215.7%**	**+160.72%**	**+105.21%**

Results for micro- and macro-f1 scores are the mean after fivefold cross-validation, with standard deviations. Lifts over uniform, stratified and frequency predictors are presented as percentages

Bold values indicate the best for that metric

Table 5 Degree (DG), triangle count (TC) and eigenvector centrality (EC) classification results for SDNE embeddings on the ego-Facebook dataset

Feature	Classifier	Micro-f1	Macro-f1	Uniform	Strat	Freq
DG	LR	0.284 (±0.013)	0.177 (±0.008)	+53.15%	+21.0%	−5.28%
	SVM (Lin)	**0.295 (±0.020)**	0.167 (±0.012)	+59.08%	+25.69%	−1.61%
	SVM (RBF)	0.289 (±0.017)	0.142 (±0.006)	+55.85%	+23.13%	−3.61%
	NN	0.253 (±0.012)	0.187 (±0.012)	+36.43%	+7.79%	−15.62%
	NN-2	0.247 (±0.018)	**0.193 (±0.019)**	+33.2%	+5.24%	−17.62%
TC	LR	0.284 (±0.015)	0.138 (±0.011)	+99.15%	+18.87%	−6.13%
	SVM (Lin)	0.296 (±0.016)	0.125 (±0.008)	+107.56%	+23.89%	−2.16%
	SVM (RBF)	**0.300 (±0.018)**	0.124 (±0.006)	**+110.37%**	+25.57%	−0.84%
	NN	0.264 (±0.020)	0.161 (±0.018)	+85.12%	+10.5%	−12.74%
	NN-2	0.247 (±0.018)	**0.162 (±0.016)**	+73.2%	+3.38%	−18.36%
EC	LR	0.297 (±0.008)	0.166 (±0.004)	+70.4%	+12.85%	−3.26%
	SVM (Lin)	**0.316 (±0.010)**	0.156 (±0.006)	**+81.3%**	**+20.07%**	**+2.93%**
	SVM (RBF)	0.309 (±0.017)	0.149 (±0.008)	+77.28%	+17.41%	+0.65%
	NN	0.286 (±0.013)	0.198 (±0.018)	+64.08%	+8.67%	−6.84%
	NN-2	0.272 (±0.018)	**0.201 (±0.014)**	+56.05%	+3.35%	−11.4%

Results for micro- and macro-f1 scores are the mean after fivefold cross-validation, with standard deviations. Lifts over uniform, stratified and frequency predictors are presented as percentages

Bold values indicate the best for that metric

prediction across all classes, all the classification results in Sect. 5 are presented using a single hidden layer neural network.

5 Results

This section presents both the supervised and unsupervised results for predicting topological features from graph embeddings.

5.1 Topological Feature Prediction

In this section, we present the experimental evaluation of the classification of topological features using the embeddings generated from the five approaches (DeepWalk, Node2Vec-H, Node2Vec-S, SDNE and PD) on the datasets detailed in Table 3. We present both the macro-f1 and micro-f1 scores plotted against a varying amount of labelled data available during the training process, where a higher score equates to

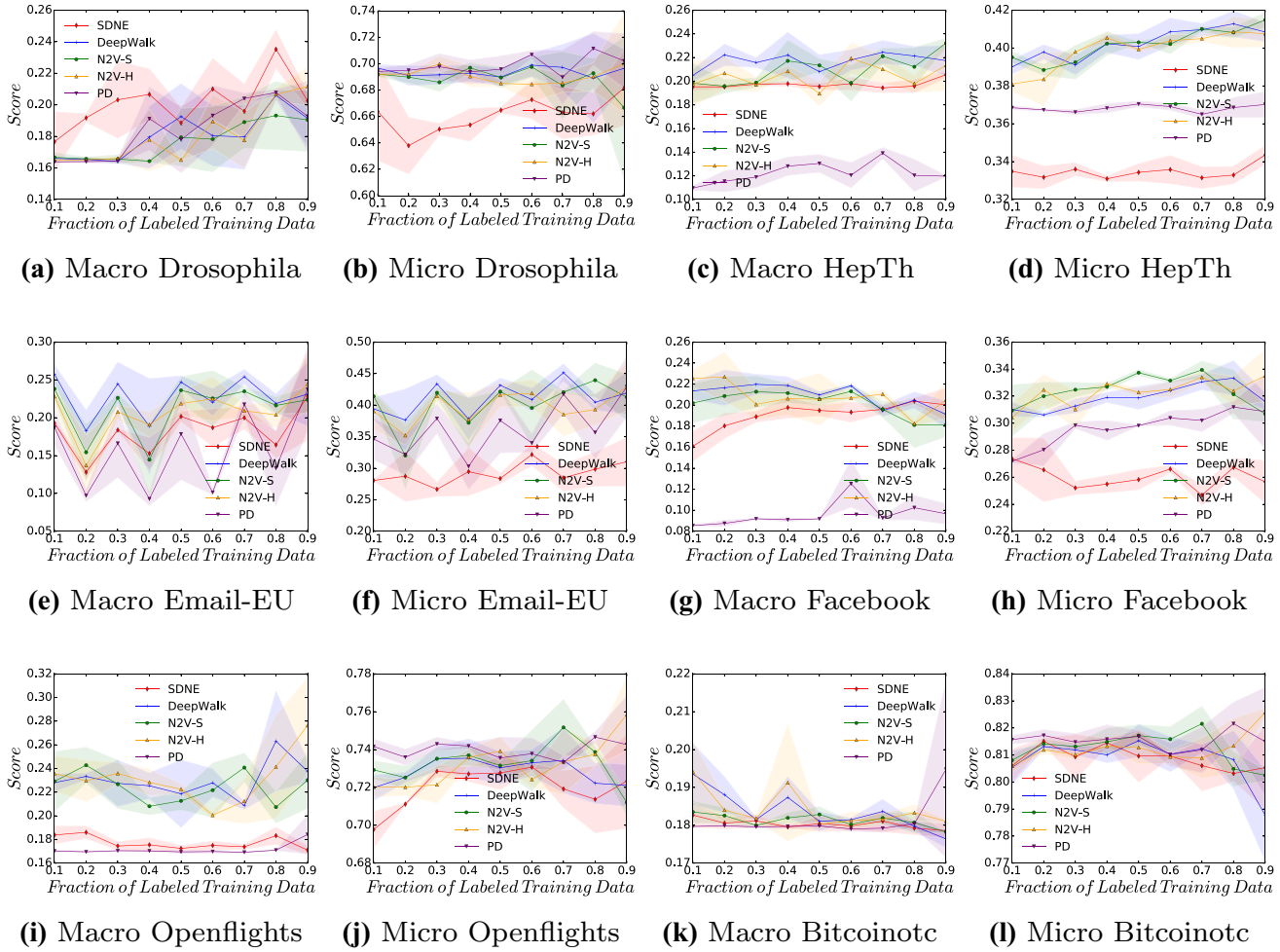


Fig. 2 Micro- and macro-f1 scores, across a range of labelling fractions, for all approaches when predicting a vertex's degree (DG) value across all datasets

a better classification result—with a score of one meaning a perfect classification of every example in the data.

Figure 2 displays the classification of f1 scores for predicting the simplest feature we are measuring: the degree of the vertices. Interestingly, we see a large spread of results across the datasets and between approaches, with no clear pattern emerging in this figure. On certain datasets, it is possible to see a high micro-f1 score, for example in the Bitcoinotc dataset, suggesting that an approximation of the degree value is present in the embedding. The figure also shows that SDNE and PD often have a lower score when compared to the stochastic approaches.

Figure 3 highlights the macro-f1 and micro-f1 scores for the classification of the degree centrality value. As the degree centrality of a given vertex is strongly influenced by its degree, it is perhaps unsurprising to observe largely similar patterns to those in Fig. 2, which again shows the dataset Bitcoinotc to be the dataset with the highest accuracies. As seen in the previous figure, generally the three stochastic approaches have a similar score for both macro-f1 and micro-f1.

The results for the classification of triangle count for the vertices are presented in Fig. 4. This is a more complex feature than the previous two, as it requires more information than is available from just the immediate neighbours of a given vertex. The figure shows again that, to some degree of accuracy, the feature is able to be reconstructed from the embedding space, with Bitcoinotc having the highest micro-f1 accuracy of all the datasets. SDNE and PD continue to have, on average, the lowest accuracies.

Classifying a vertex's local clustering score across the datasets is explored in Fig. 5. The figure shows that this feature, although more complicated to compute than a vertex's triangle count, appears to be easier for a classifier to reconstruct from the embedding space. With this more complicated feature, some interesting results regrading SDNE can be seen in the Email-EU and HepTh datasets, where the approach has the highest macro-f1 score—perhaps indicating

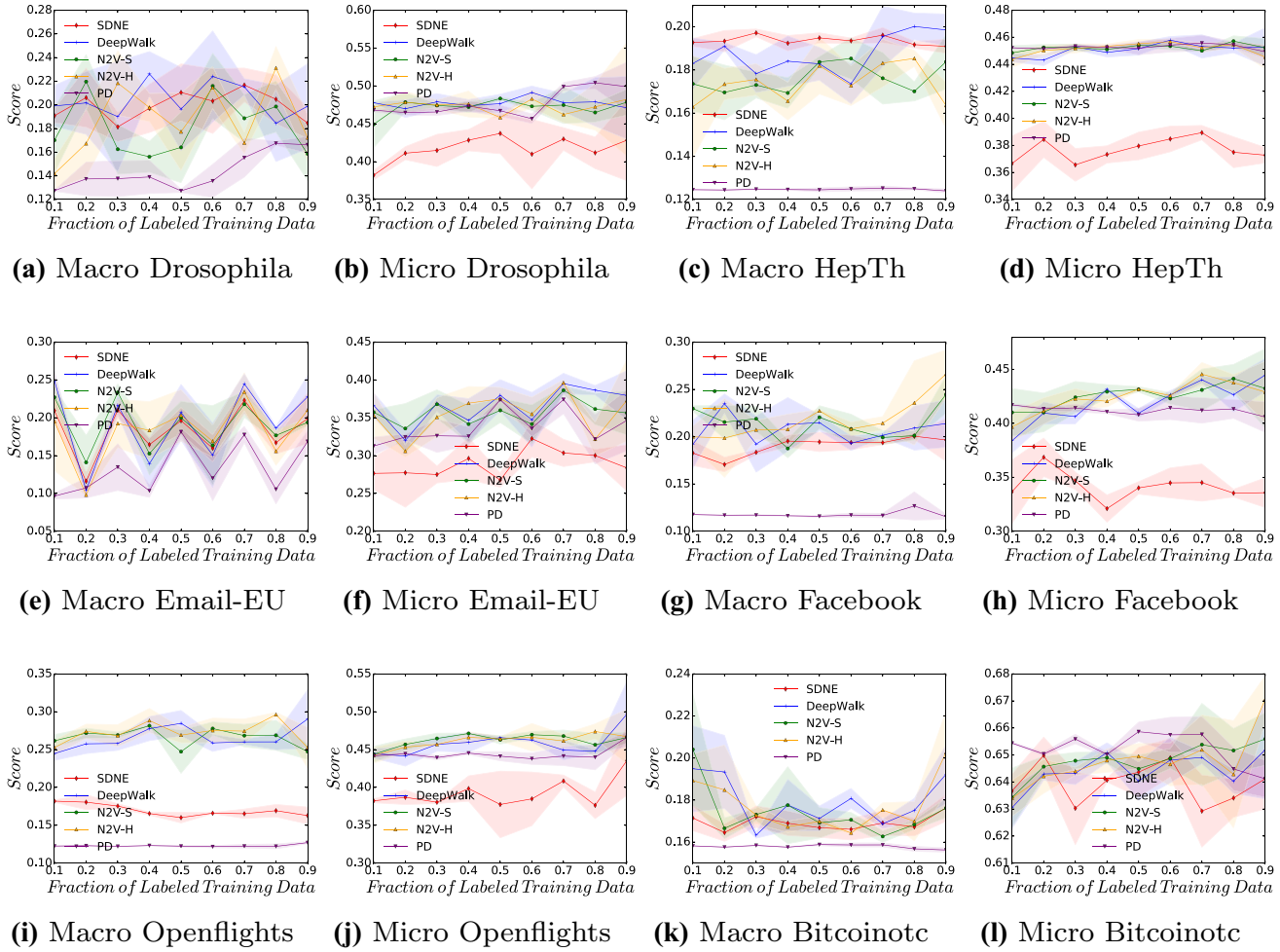


Fig. 3 Micro- and macro-f1 scores, across a range of labelling fractions, for all approaches when predicting a vertex's degree centrality (DG) value across all datasets

that the more complex model is better able to learn a good representation for this more complicated feature.

Figure 6 displays the result for the classification of a vertex's eigenvector centrality. This figure is perhaps the most interesting one so far as it shows high classification accuracies across many of the empirical datasets, even though this feature is of greater complexity than previous ones. This figure further supports the results presented in Table 4, which shows eigenvector centrality having not only the highest accuracies, but also the highest lifts in accuracy over the rule-based predictors. Interestingly, SDNE does not demonstrate higher macro-f1 scores in this experiment.

In Fig. 7, the approach's ability to correctly classify the PageRank score of the vertices is considered. Here we see generally lower classification accuracies than the last figure, perhaps owing to the more complicated nature of the PageRank algorithm. However, high classification accuracies can still be seen, particularly on the Bitcoinotc and Drosophila datasets.

Finally, Fig. 8 highlights the ability of the graph embeddings to predict betweenness centrality. Here, the figure shows that this feature is, on average, harder to predict from the embeddings than the previous two centrality measures as evidenced by the lower accuracy scores. Again, SDNE shows the highest macro-f1 scores on the Drosophila and HepTh datasets, indicating its embedding captures something akin to this structural information better than the other approaches.

5.2 Confusion Matrices

One consideration that must be made is that the binning process, used to transform the features into targets for classification, removes the inherent ordering present in continuous values. As an example, a vertex with a degree of 8 would still be classified incorrectly if the prediction was 10 or 100, but clearly one is more incorrect than the other. To address this, we present a selection of error matrices, to explore how

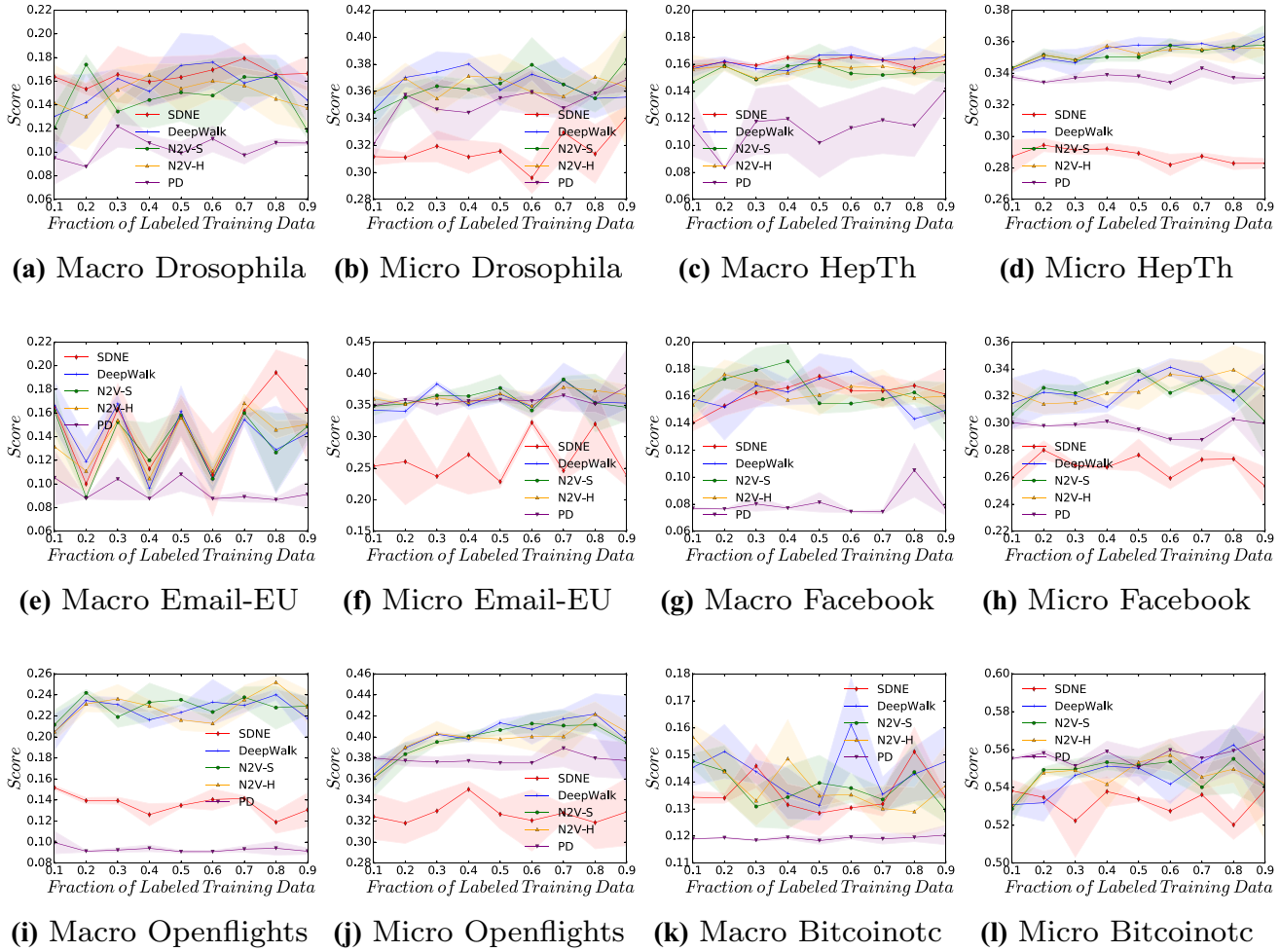


Fig. 4 Micro- and macro-f1 scores, across a range of labelling fractions, for all approaches when predicting a vertex's triangle count (TR) value across all datasets

'wrong' an incorrect prediction is. This is made possible as the labels used for classification have consecutive ordering, as a result of a histogram binning function, meaning that a prediction of 2 for a true label of 1 is more correct than a prediction of 5.

For brevity, Fig. 9 displays the error matrices for a selection of the tested embedding approaches when classifying eigenvector centrality in the ego-Facebook dataset, although similar patterns were found across all datasets. With error matrices, the diagonal values represent correctly classified label; thus, a good prediction will produce an error matrix with a higher concentration of diagonal values. Figure 9 shows that, for the stochastic walk approaches DeepWalk and Node2Vec, the error matrices have a higher clustering of values around the diagonals. Interestingly, when the classification is incorrect for these approaches, the incorrect prediction tends to be close to the true label. This phenomenon can clearly be seen in these approaches for labels 1 and 2, meaning that embeddings for vertices with these particularly eigenvector centrality are similar. The figure also shows that, for this particular vertex feature, the embeddings produced via SDNE seemingly do not contain the same topological information. This is highlighted by the lack of structure on the diagonals of its error matrix.

5.3 Unsupervised Low-Dimensional Projections

Figure 10 displays a selection of t-SNE plots taken from the ego-Facebook data, where the points are coloured according to the eigenvector centrality value after being passed through the binning process. The figure shows that the SDNE embeddings seemingly have no clear structure in the low-dimensional space which correlates strongly with the eigenvector centrality, as points in the same class are not clustered together. However, with the other embedding approaches, it is possible to see a clear clustering of points belonging to the same class. For example, in both the Node2Vec approaches, there is very clear clustering of classes one, four and five.

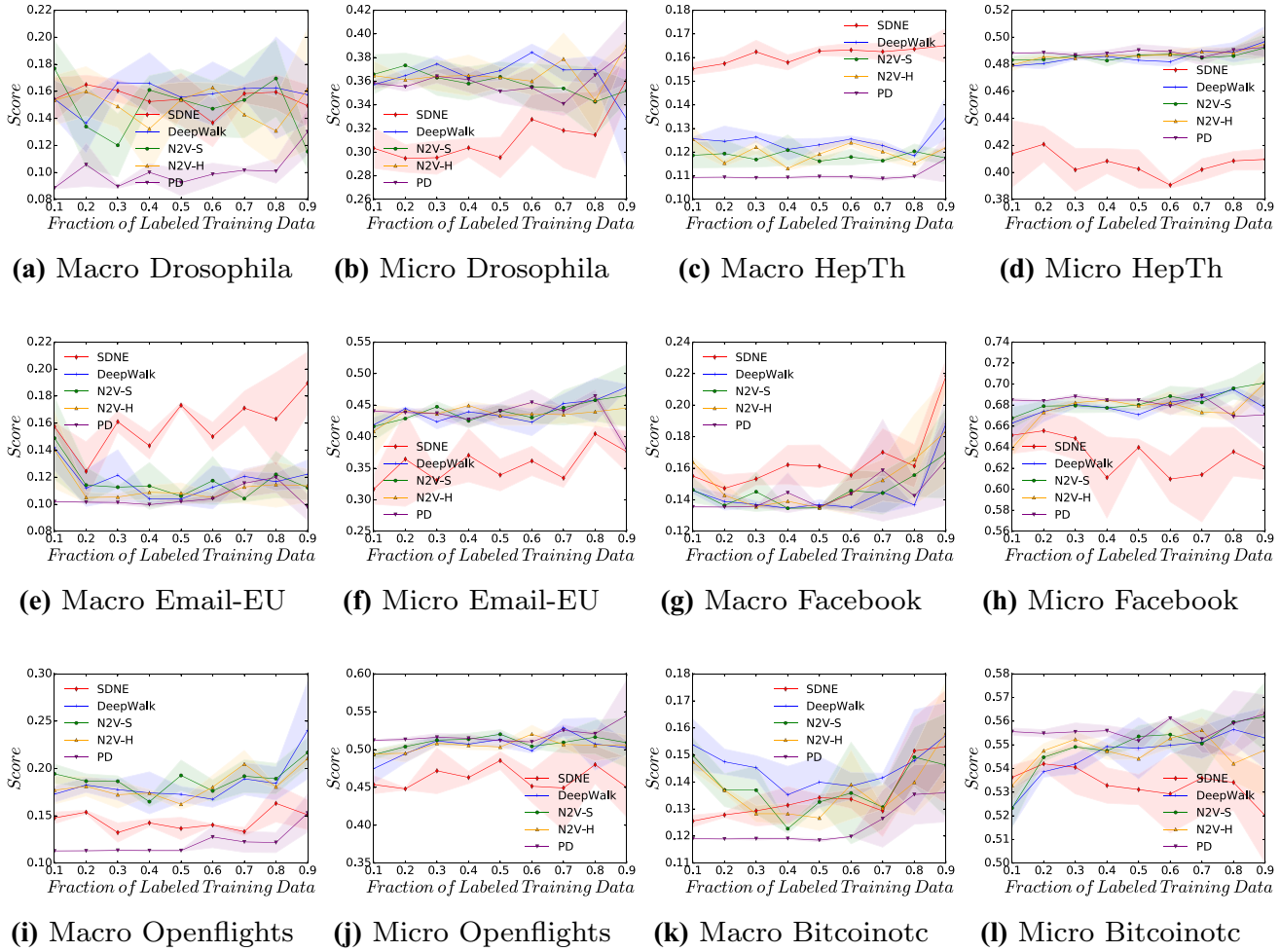


Fig. 5 Micro-f1 and macro-f1 scores, across a range of labelling fractions, for all approaches when predicting a vertex's local clustering coefficient (CLU) value across all datasets

This result provides further evidence for our observation that, even when exploring the embeddings using an unsupervised method, it is possible to find correlations between known topological features and the embedding space.

5.4 Auto-encoder Comparison

The results presented thus far have shown that it can be comparatively challenging to recover evidence of topological features from the auto-encoder-based SDNE approach. To investigate this further, we compare SDNE with another auto-encoder-based approach entitled DNGR. Unlike the other approaches tested thus far, DNGR mandates the use of weighted graphs. However, from the empirical datasets we are using for this study, only the soc-sign-bitcoinotc dataset contains weighted edges, which represent the level of trust which users place in each other.

To investigate if DNGR captures more recognisable topological structure in its embedding space, we will again use t-SNE. However, the soc-sign-bitcoinotc dataset has the lowest edge density of any of the graphs we are testing, resulting in a very unbalanced dataset. (For example, the majority of the vertices have a very low degree value.) To allow for greater insight, here we chose not to use the binning process to label each vertex embedding. Instead, we normalise the raw topological feature values to be between zero and one; we then use this value to directly colour the points on the t-SNE plots. Here we would expect to see points of a similar colour, and thus feature value, to be clustered together if vertices with similar topological features are close in the underlying embedding space. Due to soc-sign-bitcoinotc having a larger number of vertices than the dataset used for the previous t-SNE visualisation, we plot only a randomly selected half of the vertices to allow for clearer figures.

Figure 11 displays the t-SNE plots of the vertex embeddings for both SDNE and DNGR across four different topological features. The figure shows that despite it being more challenging to recover topological features from

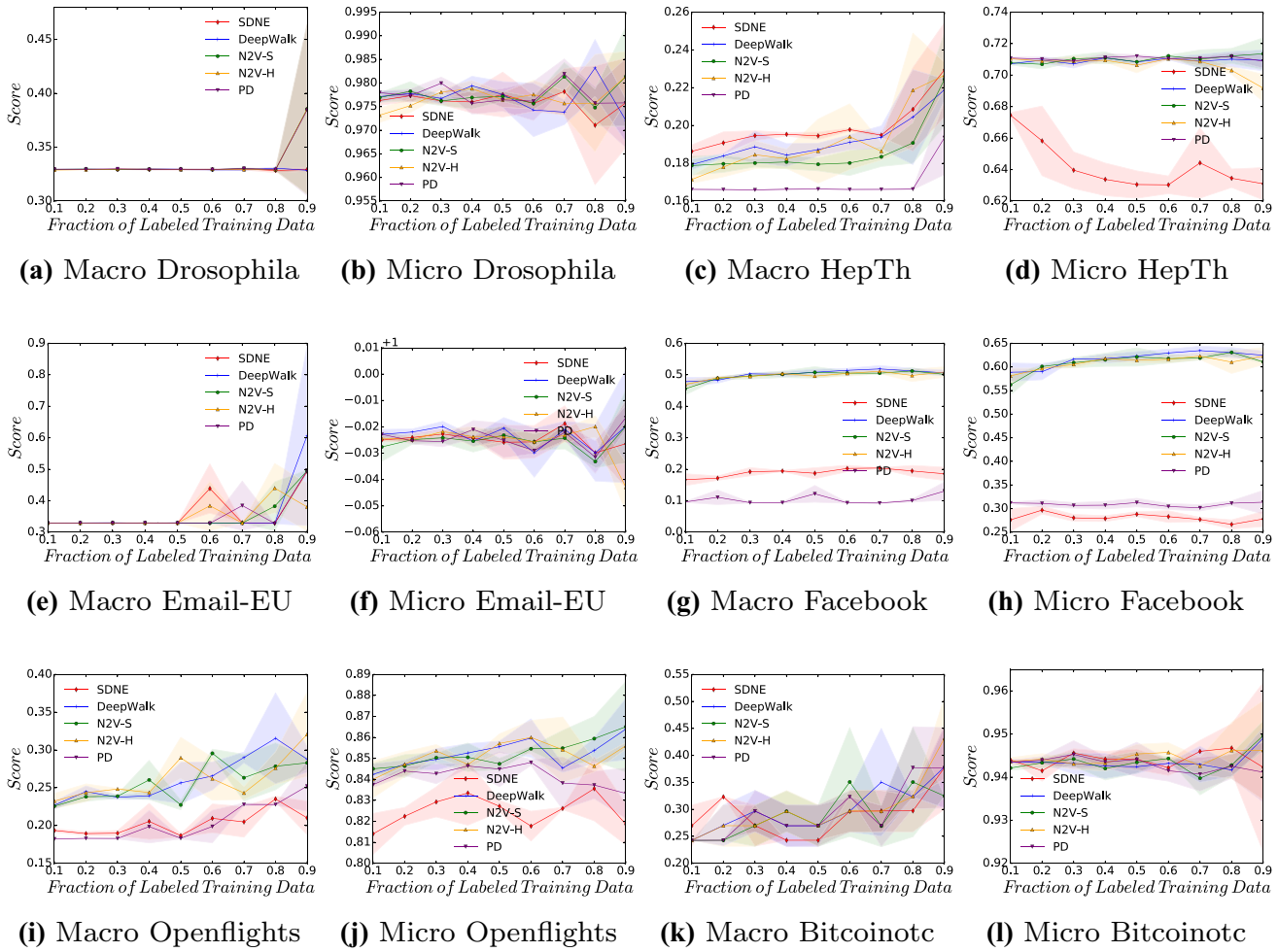


Fig. 6 Micro-f1 and macro-f1 scores, across a range of labelling fractions, for all approaches when predicting a vertex's eigenvector centrality (EC) value across all datasets

SDNE in other experiments, there is still structure present in the embedding space correlating to several topological features. One can see SDNE embeddings with similar feature values being clustered together in these plots; for example, there are clear clusters of vertices with a high and low degree, PageRank and betweenness centrality value visible, whereas it is much harder to interpret any structure in the embedding space produced via DNGR. This could well be due to the fact that DNGR does not take as input the raw adjacency matrix; instead, it is reconstructing the PPMI matrix, capturing vertex co-occurrence. Due to this transformed input, it is perhaps not surprising that normal topological features are present in the resulting representations.

5.5 Discussion

This section has provided extensive experimentation evaluation to explore the questions raised in Sect. 3. Specifically, we investigated if a broad range of topological features can be predicted from the embedding created from a range of unsupervised graph embedding techniques. Across all the features and datasets tested, it can be seen that many topological features can be approximated by the different embedding approaches, with varying degrees of accuracy. The results which show the increase in accuracy over the rule-based predictions (Sect. 4.3) give strong indication that the approaches are able to overcome the inherent unbalanced nature of graph datasets and a mapping from the embedding space to features is present. It is also interesting to observe that numerous features can be approximated from the graph embeddings, suggesting that several structural properties are being captured to create the best representation for a vertex automatically. Of all the topological features measured in the experimentation section, the one which consistently gave the best results was eigenvector centrality. Particularly for the stochastic approaches, eigenvector centrality was predicted with a high degree of accuracy, suggesting that the

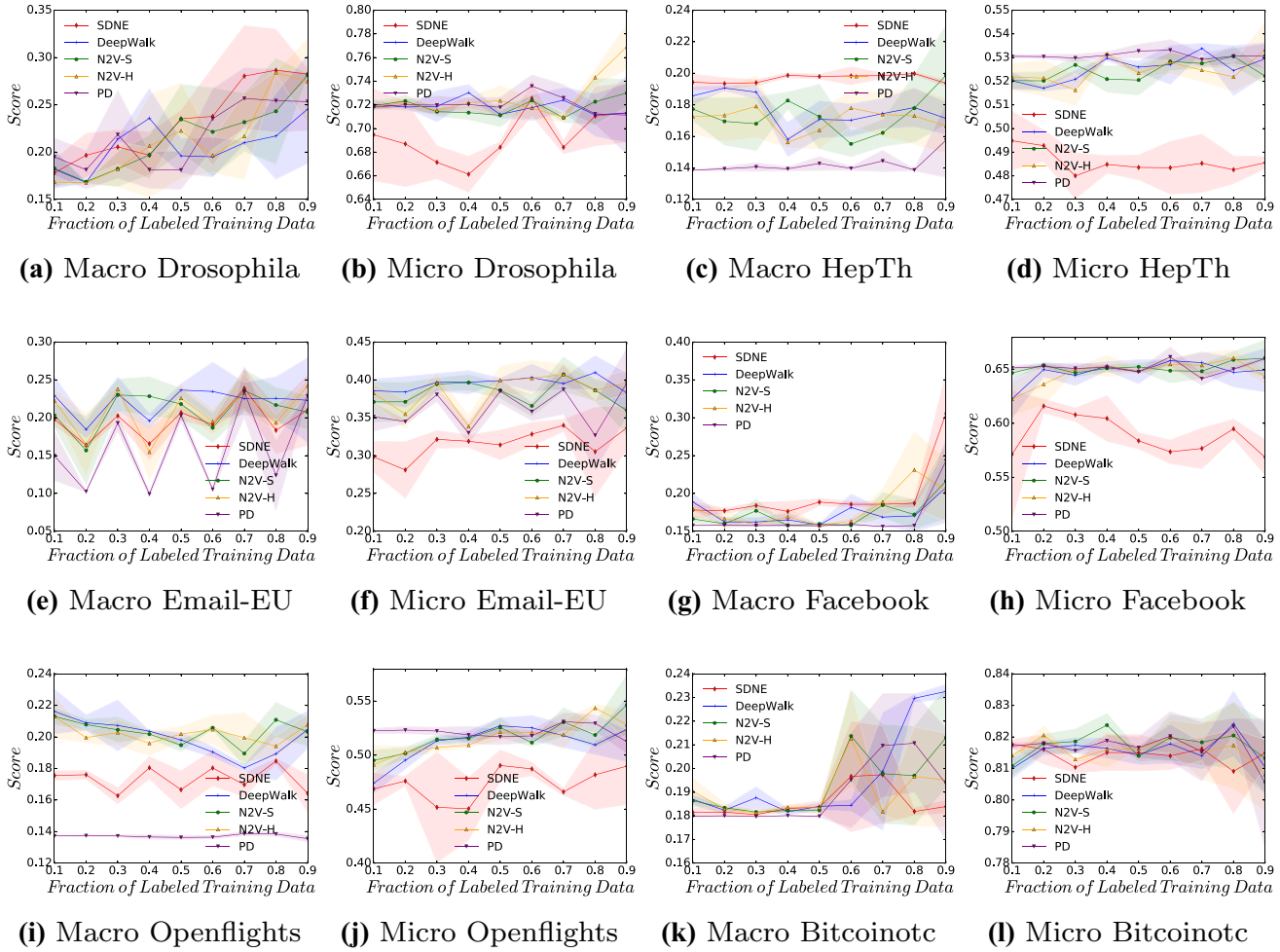


Fig. 7 Micro-f1 and macro-f1 scores, across a range of labelling fractions, for all approaches when predicting a vertex's PageRank (PR) value across all datasets

topological structure represented by this feature is captured extremely well in the embedding space and indicates this is a useful feature for minimising the objective functions of the approaches. This is further reinforced by the unsupervised projections (Fig. 10), which shows clear and distinct clustering between classes, even without the use of a classification algorithm.

Another interesting observation from this study is that no one approach strongly outperforms the others when classifying a particular feature—seemingly all the approaches are approximating similar topological structures. The figures show that the stochastic approaches (DeepWalk and Node2Vec) are the most consistent across all features and datasets, often having the highest macro-f1 and micro-f1 scores. SDNE demonstrates a more inconsistent performance profile for feature classification; this is in contrast to other studies which have found it to have the best performance in vertex labelling problems [2]. The performance of SDNE demonstrated in this work could be explained by it being the only deep model tested, meaning that it contains many more parameters. This increase in complexity means that SDNE could be very sensitive to the correct selection of hyper-parameters or possibly that more complex topological features are being approximated by the embeddings—or even that entirely novel features are being learned. Finally, it is interesting to note the performance of hyperbolic (PD) approach, which has far fewer latent dimensions in which to capture topological information due to its limitation in modelling the space as a 2D disc. Empirically, PD shows largely similar performance to the other approaches on most datasets, providing strong evidence that the hyperbolic space is an appropriate space in which to represent graphs.

6 Conclusion

Graph embeddings are increasingly becoming a key tool to solve numerous tasks within the field of graph mining. They have demonstrated state-of-the-art results by reporting to

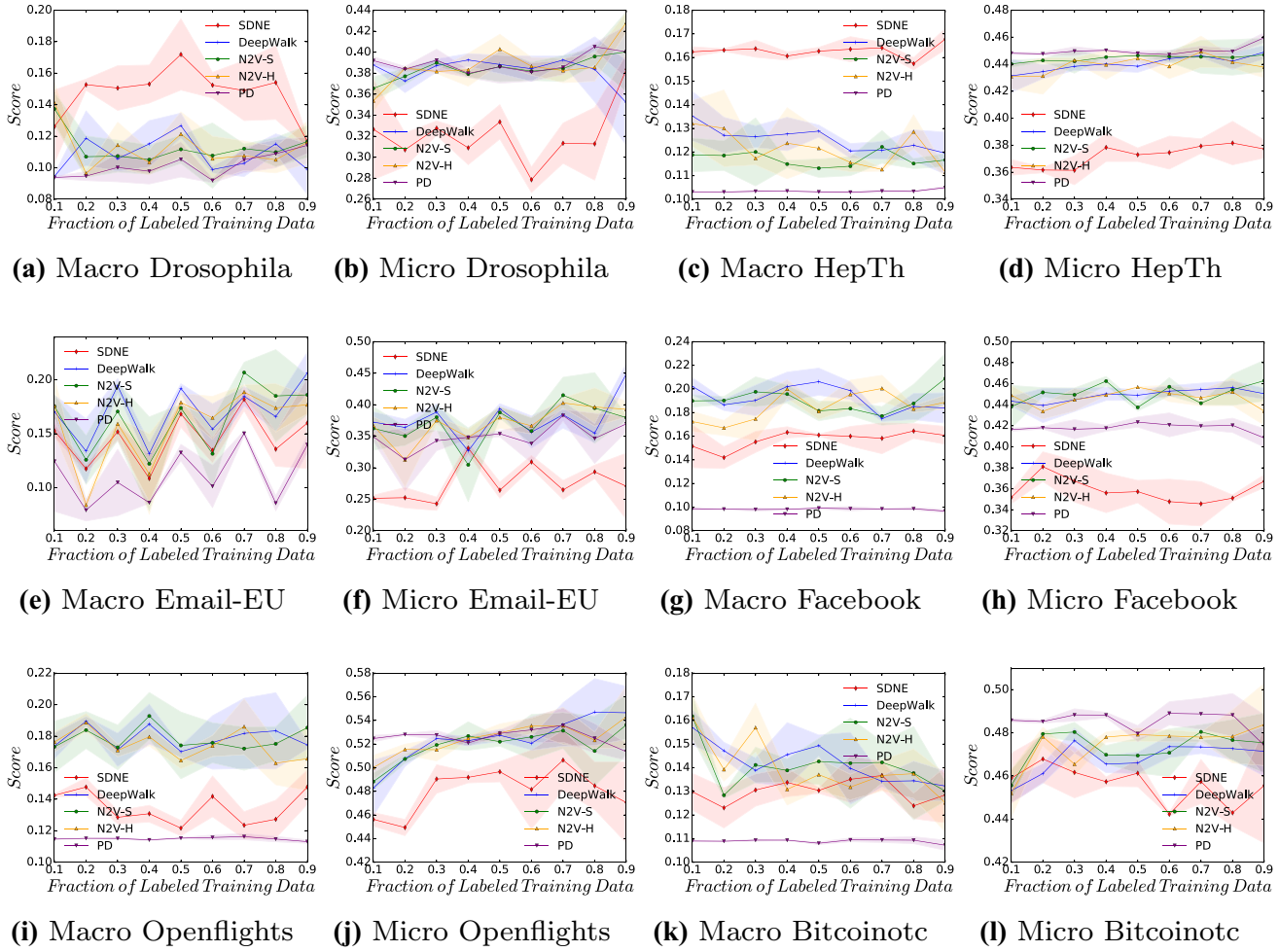


Fig. 8 Micro-f1 and macro-f1 scores, across a range of labelling fractions, for all approaches when predicting a vertex's betweenness centrality (BC) value across all datasets

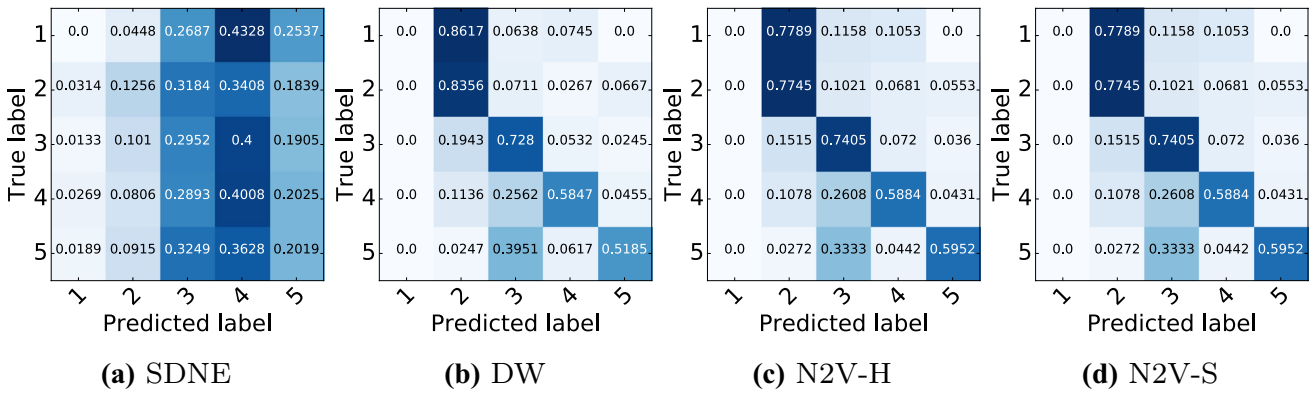


Fig. 9 Error matrices for neural network classification of eigenvector centrality (EC) for the ego-Facebook dataset

automatically learn a low-dimensional, but highly expressive, representation of vertices, which captures the topological structure of the graph. However, to date, there has been little work providing a theoretical grounding which would allow for greater interpretability. In this paper, we explore making a step in this direction by investigating which

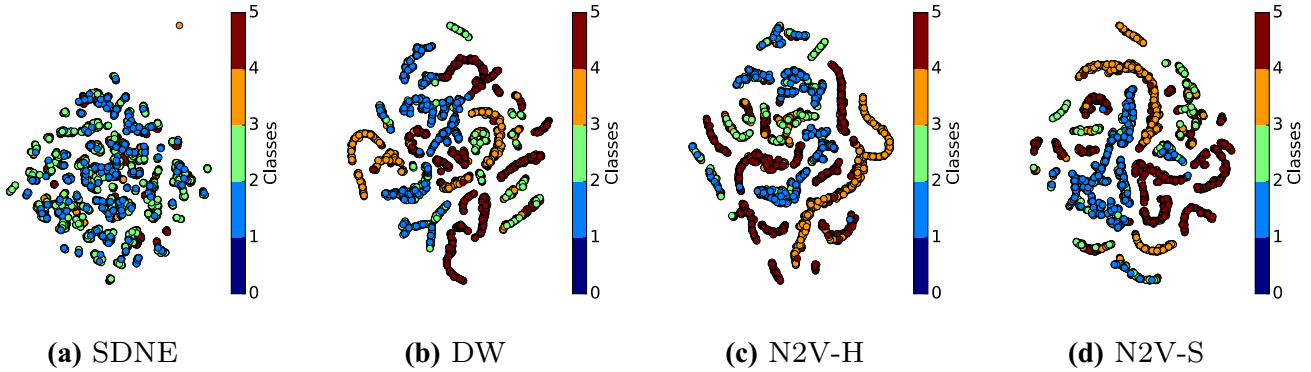


Fig. 10 t-SNE plots of the embeddings taken from the ego-Facebook dataset, where the points are coloured according to their eigenvector centrality (EC)

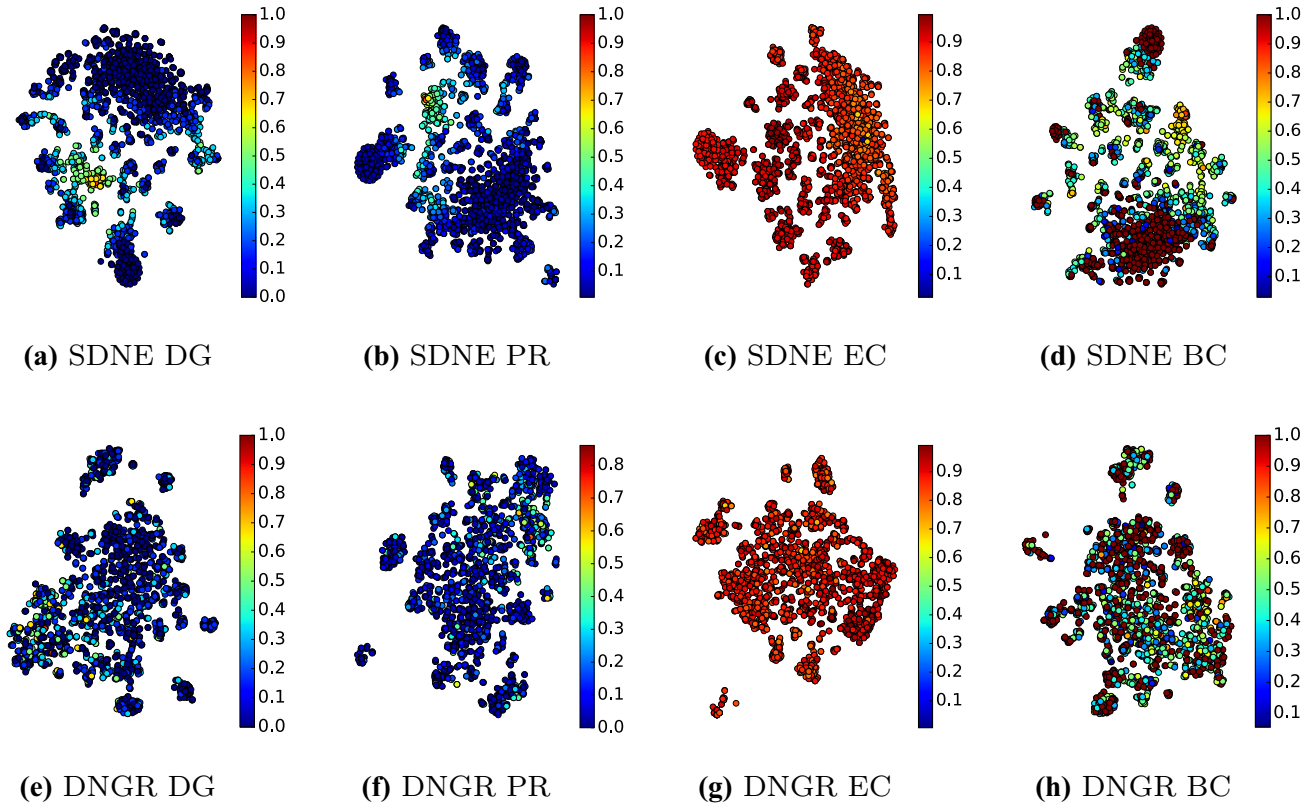


Fig. 11 t-SNE plots of SDNE and DNGR embeddings taken from the soc-sign-bitcoinotc dataset, where points are coloured according to the normalised topological feature

traditional topological graph features can be reconstructed from the embedding space, the hypothesis being that if a mapping from the embedding space to a particular topological feature can be found, then the topological structure encapsulated by this feature is also captured by the embedding. We present an extensive set of experiments exploring this issue across five unsupervised graph embedding techniques (detailed in Sect. 3.4), classifying seven graph features (detailed in Sect. 3.1), across a range of empirical datasets (detailed in Table 3). We find that a mapping from many topological features to the embedding space of the tested approaches is indeed possible, using both supervised and unsupervised techniques. This discovery suggests that graph embeddings are indeed learning approximations of known topological features, with our experiments showing that eigenvector centrality is best reconstructed by many of

the approaches. This could allow key insight into how graph embedding learn to create high-quality representations.

For future research, we plan to see if other eigenvector-based topological features, known to be representative of a graph's topology [9], are also captured as well by the embedding approaches. We plan to perform more experimentation with synthetically created graphs with artificially balanced degree distributions. This will remove the unbalanced nature of empirical datasets and allow us to explore the structure of the embeddings in more detail. Furthermore, we plan to investigate if directly predicting topological features during the embedding training process, perhaps in the form of a regularisation term, can produce embeddings which generalise better across other tasks.

Acknowledgements We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research. Additionally, we thank the Engineering and Physical Sciences Research Council UK (EPSRC) for funding. For invaluable feedback and comments during this research, we also thank Nik Khadijah Nik Aznan, Philip Jackson and Amir Atapour-Abarghouei. We also thank the authors of papers [5, 32, 37] for making implementations of their code publicly available.

References

1. Newman M (2010) Networks: an introduction. Oxford University Press, Oxford
2. Goyal P, Ferrara E (2017) Graph embedding techniques, applications, and performance: a survey. arXiv preprint arXiv:1705.02801
3. Moyano LG (2017) Learning network representations. Eur Phys J Spec Top 226(3):499–518
4. Perozzi B, Al-Rfou R, Skiena S (2014) DeepWalk: online learning of social representations. In: ACM SIGKDD international conference on knowledge discovery and data mining
5. Grover A, Leskovec J (2016) node2vec : scalable feature learning for networks. In: ACM SIGKDD international conference on knowledge discovery and data mining
6. Bonner S, Brennan J, Theodoropoulos G, Kureshi I, McGough AS, Obara B (2017) Evaluating the quality of graph embeddings via topological feature reconstruction. In: IEEE international conference on big data
7. Obara B, Grau V, Fricker MD (2012) A bioimage informatics approach to automatically extract complex fungal networks. Bioinformatics 28(18):2374
8. Page L, Brin S, Motwani R, Winograd T (1999) The PageRank citation ranking: bringing order to the web. Stanford InfoLab
9. Li G, Semerci M, Yener B, Zaki MJ (2012) Effective graph classification based on topological and label attributes. Stat Anal Data Min ASA Data Sci J 5(4):265
10. Bonner S, Brennan J, Theodoropoulos G, Kureshi I, McGough AS (2016) Deep topology classification: a new approach for massive graph classification. In: IEEE international conference on big data
11. Berlingerio M, Koutra D, Eliassi-Rad T, Faloutsos C (2012) NetSimile: a scalable approach to size-independent network similarity. arXiv preprint arXiv:1209.2684
12. Bonner S, Brennan J, Theodoropoulos G, Kureshi I, McGough AS (2016) Gfp-x: a parallel approach to massive graph comparison using spark. In: IEEE international conference on big data, pp 3298–3307
13. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3(Mar):1157
14. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
15. Hamilton WL, Ying R, Leskovec J (2017) Representation learning on graphs: methods and applications. arXiv preprint arXiv:1709.05584
16. Cai H, Zheng VW, Chang KCC (2017) A comprehensive survey of graph embedding: problems, techniques and applications. arXiv preprint arXiv:1709.07604
17. Zhang D, Yin J, Zhu X, Zhang C (2017) Network representation learning: a survey. arXiv preprint arXiv:1801.05852
18. Cui P, Wang X, Pei J, Zhu W (2017) A survey on network embedding. arXiv preprint arXiv:1711.08752
19. Bruna J, Zaremba W, Szlam A, LeCun Y (2013) Spectral networks and locally connected networks on graphs. In: International conference on learning representations (ICLR)
20. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems (NIPS)
21. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: International conference on learning representations (ICLR)
22. Niepert M, Ahmed M, Kutzkov K (2016) Learning convolutional neural networks for graphs. In: International conference on machine learning
23. Wold S, Esbensen K, Geladi P (1987) Principal component analysis. Chemometr Intell Lab Syst 2(1–3):37
24. Belkin M, Niyogi P (2002) Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in neural information processing systems, pp 585–591
25. Ahmed A, Shervashidze N, Narayanamurthy S, Josifovski V, Smola AJ (2013) Distributed large-scale natural graph factorization. In: International conference on World Wide Web, pp 37–48
26. Cao S, Lu W, Xu Q (2015) Grarep: learning graph representations with global structural information. In: ACM international on conference on information and knowledge management, pp 891–900
27. Ou M, Cui P, Pei J, Zhang Z, Zhu W (2016) Asymmetric transitivity preserving graph embedding. In: ACM SIGKDD international conference on knowledge discovery and data mining, pp 1105–1114
28. Mikolov T, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Conference on neural information processing systems (NIPS)
29. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: International conference on learning representations (ICLR)
30. Backstrom L, Leskovec J (2011) Supervised random walks: predicting and recommending links in social networks. In: Web search and data mining (WSDM)
31. Nickel M, Kiela D (2017) Poincaré embeddings for learning hierarchical representations. arXiv preprint arXiv:1705.08039
32. Chamberlain B, Clough J, Deisenroth MP (2017) Neural embeddings of graphs in hyperbolic space. In: KDD workshop on mining and learning with graphs (MLG)

33. Munzner T (1998) Exploring large graphs in 3D hyperbolic space. In: IEEE computer graphics and applications
34. Epstein DB, Penner RC et al (1988) Euclidean decompositions of noncompact hyperbolic manifolds. J Differ Geomet 27(1):67–80
35. Hinton GE, Krizhevsky A, Wang SD (2011) Transforming auto-encoders. In: International conference on artificial neural networks
36. Salakhutdinov R, Hinton G (2009) Semantic hashing. Int J Approx. Reason 50(7):969–978
37. Wang D, Cui P, Zhu W (2016) Structural deep network embedding. In: ACM SIGKDD international conference on knowledge discovery and data mining
38. Cao S, Lu W, Xu Q (2016) In: 30th AAAI conference on artificial intelligence
39. Erhan D, Bengio Y, Courville A, Manzagol PA, Vincent P, Bengio S (2010) Why does unsupervised pre-training help deep learning? J Mach Learn Res 11:625–660
40. Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs. arXiv preprint arXiv:1706.02216
41. Li C, Guo X, Mei Q (2016) Deepgraph: graph structure predicts network growth. arXiv preprint arXiv:1610.06251
42. Liu W, Cooper H, Oh MH, Yeung S, Chen Py, Suzumura T, Chen L (2017) Learning graph topological features via GAN. arXiv preprint arXiv:1709.03545
43. Albert R, Barabási A (2002) Statistical mechanics of complex networks. Rev Modern Phys 74(1):47–97
44. Salehi Rizi F, Granitzer M, Ziegler K (2017) Properties of vector embeddings in social networks. Algorithms 10(4):109
45. Schnabel T, Labutov I, Mimno D, Joachims T (2015) Evaluation methods for unsupervised word embeddings. In: Conference on empirical methods in natural language processing, pp 298–307
46. Li J, Chen X, Hovy E, Jurafsky D (2015) Visualizing and understanding neural models in NLP. arXiv preprint arXiv:1506.01066
47. Conneau A, Kiela D, Schwenk H, Barrault L, Bordes A (2017) Supervised learning of universal sentence representations from natural language inference data. arXiv preprint arXiv:1705.02364
48. Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: European conference on computer vision, pp 818–833
49. Watts DJ, Strogatz SH (1998) Collective dynamics of 'small-world' networks. Nature 393:440–442
50. Bonacich P (2007) Some unique properties of eigenvector centrality. Soc Netw 29(4):555
51. Han M, Daudjee K, Ammar K, Ozsu MT, Wang X, Jin T (2014) An experimental comparison of pregel-like graph processing systems. VLDB Endowment 7(12):1047
52. Faloutsos M, Faloutsos P, Faloutsos C (1999) On power-law relationships of the internet topology. In: ACM SIGCOMM computer communication review
53. Oord Avd, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K (2016) Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499
54. Lvd Maaten, Hinton G (2008) Visualizing data using t-sne. J Mach Learn Res 9(Nov):2579
55. Arlot S, Celisse A (2010) A survey of cross-validation procedures for model selection. Stat Surv 4:40–79
56. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, et al (2016) Tensorflow: a system for large-scale machine learning. In: USENIX symposium on operating systems design and implementation, vol 16, p 265
57. Shi S, Wang Q, Xu P, Chu X (2016) Benchmarking state-of-the-art deep learning software tools. arXiv preprint arXiv:1608.07249
58. Leskovec J, Krevl A (2014) SNAP datasets: Stanford large network dataset collection. http://snap.stanford.edu/data. Accessed Feb 2018
59. Rossi RA, Ahmed NK (2015) The network data repository with interactive graph analytics and visualization. In: AAAI conference on artificial intelligence. http://networkrepository.com. Accessed Feb 2018
60. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
61. Karakoulas GI, Shawe-Taylor J (1999) Optimizing classifers for imbalanced training sets. In: Advances in neural information processing systems, pp 253–259